**Hand length measurements**

1. **Flowchart:**

Yes

Load the hand detector Initialize camera

Is Camera connected

Get Height and width of frame from the camera

Identify the Hand

If Hand found?

Detect Hand using openCV

Is the image displayed?

Display RGB image with the length of the Hand

Is Hand found?

Display Joining points on Hand

HA

Extract required points from the “lmList” from CVzone

Convert points in pixels to cm

Get the distance between the points using distance between 3D vector formula

Yes

No

No

No

No

Yes

Yes

1. **Algorithm:**

**Step1:** Start

**Step2:** Import the necessary libraries

**Step3:** Initialize the cv2, OpenCv and CVzone for Hand module and the webcam capture. If webcam is not detected the jump to Step 9

**Step4:** Set up the main loop for capturing and processing frames from the webcam

**Step5:** Check if hand landmarks are detected in the frame and store it in the “lmList”. If Hand is not detected then jump to Step 9

**Step6:** Access the landmarks of the thumb, middle, little finger and wrist. If landmarks can’t able to access then Jump to Step 9

**Step7:** Convert the landmarks to pixel coordinates using Distance between 3D vector formula. Repeat the procedure from Step 5 whenever the Hand are detected in the frame.

**Step8:** Display the land mark with joining points in the frame

**Step9:** Exit

1. **Software Used:**

Pycharm – Python Programming

By using OpenCV and CVzone

1. **Code:**

import math

import cv2

from cvzone.HandTrackingModule import \*

import numpy as np

import cvzone

import mediapipe as mp

#setup

mp\_drawing = mp.solutions.drawing\_utils

mp\_drawing\_styles = mp.solutions.drawing\_styles

mp\_pose = mp.solutions.pose

#webcam

cap = cv2.VideoCapture(0)

cap.set(3, 1288)

cap.set(4, 720)

#Hand and pose detector

detector = HandDetector(detectionCon=0.8, maxHands=2, minTrackCon=0.5 )

#loop

while True:

success, img = cap.read()

hands, img = detector.findHands(img)

if hands:

lmList = hands[0]['lmList']

x, y, w, h = hands[0]['bbox']

x0, y0, z0 = lmList[5]

x1, y1, z1 = lmList[17]

x2, y2, z2 = lmList[0]

x3, y3, z3 = lmList[12]

d1 = math.sqrt((x0 - x1) \*\* 2 + (y0 - y1) \*\* 2 + (z0 - z1) \*\* 2 )/17.16

d2 = math.sqrt((x2 - x3) \*\* 2 + (y2 - y3) \*\* 2 + (z2 - z3) \*\* 2 )/17.16

# displaying values

cvzone.putTextRect(img, f'{int(d1)} cm', (x+200, y))

cvzone.putTextRect(img, f'{int(d2)} cm', (x+200, y - 50))

print(int(d1),int(d2))

cv2.imshow("Image", img)

cv2.waitKey(1)

1. **Maximum and Minimum distance from the camera**

**Arm length Measurement**

1. **Flowchart:**

Load Pose detector Initialize camera

Is Camera connected?

Get Height and width of frame from the camera

Identify the Pose Module

If Pose found?

Detect the pose using mediapipe

Is the image displayed?

Display RGB image with the length of the right/left arm

Is Pose found?

Display points using openCV

HA

Extract required points from the “lmList” using mediapipe

Convert points in pixels to cm

Get the distance between the points using distance between 3D vector formula

Yes

No

No

No

Yes

No

Yes

Yes

1. **Algorithm:**

**Step1:** Start

**Step2:** Import the necessary libraries

**Step3:** Initialize the Mediapipe Pose module and the webcam capture. If webcam is not detected the jump to Step 9

**Step4:** Set up the main loop for capturing and processing frames from the webcam

**Step5:** Check if Arm landmarks are detected in the frame and store it in the “lmList”. If Arm is not detected then jump to Step 9

**Step6:** Access the landmarks of the elbow and wrist. If landmarks can’t able to access then Jump to Step 9

**Step7:** Convert the landmarks to pixel coordinates using Distance between 3D vector formula. Repeat the procedure from Step 5 whenever the Arm are detected in the frame.

**Step8:** Display the land mark with joining points in the frame

1. **Software Used:**

Pycharm – Python Programming

By using OpenCV, CVzone, Mediapipe.

1. **Code:**

import math

import cv2

import numpy as np

import cvzone

import mediapipe as mp

import cvzone.PoseModule as pm

#webcam

cap = cv2.VideoCapture(0)

cap.set(3, 1288)

cap.set(4, 720)

#pose detector

detector = pm.PoseDetector()

#setup

mp\_drawing = mp.solutions.drawing\_utils

mp\_drawing\_styles = mp.solutions.drawing\_styles

mp\_pose = mp.solutions.pose

#loop

while True:

success, img = cap.read()

img = detector.findPose(img,draw=False)

lmList = detector.findPosition(img,draw=False)

if len(lmList) != 0:

h0, x0, y0, z0 = detector.lmList[13] #Left elbow

h1, x1, y1, z1 = detector.lmList[15] #Left Wrist

h2, x2, y2, z2 = detector.lmList[14] #Right elbow

h3, x3, y3, z3 = detector.lmList[16] #Right wrist

d1 = math.sqrt((x0 - x1) \*\* 2 + (y0 - y1) \*\* 2 + (z0 - z1) \*\* 2)/17.16

d2 = math.sqrt((x2 - x3) \*\* 2 + (y2 - y3) \*\* 2 + (z2 - z3) \*\* 2)/17.16

print(int(d1),int(d2))

cv2.rectangle(img, (0, 0), (620, 150), (245, 117, 16), -1)

#Display values of d1 and d2

cv2.putText(img, str(int(d1)),(10, 60),cv2.FONT\_HERSHEY\_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.putText(img, str(int(d2)),(10, 130),cv2.FONT\_HERSHEY\_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE\_AA)

#Display data left arm pr right arm)

cv2.putText(img, 'cm - Left arm',(90, 60),cv2.FONT\_HERSHEY\_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.putText(img, 'cm - Right arm', (90, 130), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.imshow("Image", img)

cv2.waitKey(1)

1. **Maximum and Minimum distance from the camera**